

## C1

(15, 11)汉明码的验证矩阵 $H$

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

变为标准形 $H_s$

$$H_s = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$H_s = (A|I_4)$ ，因此得到 $G_s = (I_{11} | -A^T)$

$$G_s = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

之后，交换 $H_s$ 的列到 $H$ ，对 $G_s$ 做同样的列交换，得到 $G$

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

编程实现结果

```
(root@kali) - [/home/kali/Documents/Chanelcode/MyCode]
# diff -d hamming 15 11.txt ../hamming 15 11.txt

(root@kali) - [/home/kali/Documents/Chanelcode/MyCode]
# python3 hamming 15 11.py
2048 points checked, decode test pass!!!
2048 points checked, decode( with one error) test pass!!!
```

其中解码检测分为无错误解码检测与输入有随机一错误的解码检测

```
#this part used to check decode
with open('hamming_15_11.txt', 'r') as f:
    checkpoint = 0
    for line in f:
        codeRes = line.split(' ')[1].replace('\n', '')
        ham.updateCodeRes(codeRes)
        expectedAns = line.split(',')[0]
        ans = ham.decode()
        if not ans == expectedAns:
            print('check failed in {} -> {},{} expected'.format(codeRes,ans,expectedAns))
            exit(-1)
        checkpoint += 1
    print('%d points checked, decode test pass!!!' % checkpoint)
f.close()

#this part used to simulate error and check decode
with open('hamming_15_11.txt', 'r') as f:
    checkpoint = 0
    for line in f:
        codeRes = line.split(' ')[1].replace('\n', '')
        codeRes = [int(e) for e in codeRes]
        errorP = randint(0, 14)
        codeRes[errorP] = (codeRes[errorP]+1) % 2
        codeRes = ''.join(str(e) for e in codeRes)
        ham.updateCodeRes(codeRes)
        expectedAns = line.split(',')[0]
        ans = ham.decode()
        if not ans == expectedAns:
            print('check failed in {} -> {},{} expected'.format(codeRes,ans,expectedAns))
            exit(-1)
        checkpoint += 1
    print('%d points checked, decode( with one error) test pass!!!' % checkpoint)
```

## C2

存储空间角度：(255,247)汉明码的校验矩阵大小为 $H_{8 \times 255}$ ，生成矩阵大小为 $G_{247 \times 255}$ ，会占据嵌入式设备较大的内存空间。当该设备存储资源非常紧张时，会导致内存不足

时间角度：编码需要的运算次数为247次乘法与加法。解码正确时需要的运算次数为255次乘法与加法。这可能会占用该运算资源非常紧张的嵌入式设备较多的时间，编码解码速度可能无法满足数据的输入速度，导致数据滞留至内存不足，数据丢失。

## OP1

(7,4)汉明码的生成矩阵为:

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$d = [d_0 \quad d_1 \quad d_2 \quad d_3]$ ，算得

$$dG = [d_0 + d_1 + d_3 \quad d_0 + d_2 + d_3 \quad d_0 \quad d_1 + d_2 + d_3 \quad d_1 \quad d_2 \quad d_3]$$

由奇偶校验的观点,  $p_0$  校验的位置为 1, 3, 5, 7, 因此  $p_0 = d_0 + d_1 + d_3$

$p_1$  校验的位置为 2, 3, 6, 7, 因此  $p_1 = d_0 + d_2 + d_3$

$p_2$  校验的位置为 4, 5, 6, 7, 因此  $p_2 = d_1 + d_2 + d_3$

可见, 编码时奇偶校验方法与线性代数构造方法等价

## OP2

1到7的二进制表示分别为 001, 010, 011, 100, 101, 110, 111,  $p_0 = d_0 + d_1 + d_3$  即为二进制表示最后一位为1的比特的异或结果, 注意到  $p_1, p_2, d_2$  位置二进制表示最后一位为0, 不影响  $b = b_2 b_1 b_0$  的最后一位  $b_0$

, 因此  $p_0$  可以通过所有值为1的比特位置的异或的结果得到。  $p_1, p_2$  同理

## OP3

(7,4) 汉明码的校验矩阵为

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$r = [p_0 \quad p_1 \quad d_0 \quad p_2 \quad d_1 \quad d_2 \quad d_3]$$

$$\text{计算得: } Hr^T = \begin{bmatrix} p_0 + d_0 + d_1 + d_3 \\ p_1 + d_0 + d_2 + d_3 \\ p_2 + d_1 + d_2 + d_3 \end{bmatrix}$$

1到7的二进制表示为 001, 010, 011, 100, 101, 110, 111, 因此  $b_1 = p_0 + d_0 + d_1 + d_3$ ,

$b_2 = p_1 + d_0 + d_2 + d_3$ ,  $b_3 = p_2 + d_1 + d_2 + d_3$ , 当  $b = 0$ , 则  $Hr^T = 0$ , 传输正确, 提取

$d_0 d_1 d_2 d_3$  作为结果。当  $b$  不等于 0,  $Hr^T = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$ , 因此即为位置  $b$  出错, 翻转提取结果  $d_0 d_1 d_2 d_3$

## C3

编码实现结果:

```
(root@kali) - [/home/kali/Documents/Chanelcode/MyCode]
# diff -d hamming 15 11 v2.txt ../hamming 15 11.txt

(root@kali) - [/home/kali/Documents/Chanelcode/MyCode]
# python3 hamming 15 11 v2.py
2048 points checked, decode test pass!!!
2048 points checked, decode(with one error) test pass!!!
```

(其中解码结果检测方法与C1相同)

## C4

空间上：使用奇偶校验码方法时只需要在内存中存储编码输入或者解码输入，而不需要存储生成矩阵、校验矩阵，能减少内存空间占用

时间上：使用奇偶校验方法只需要对非0位的比特位置异或，至多需要255次异或操作，而不需要矩阵操作，能加快编码与解码速度。

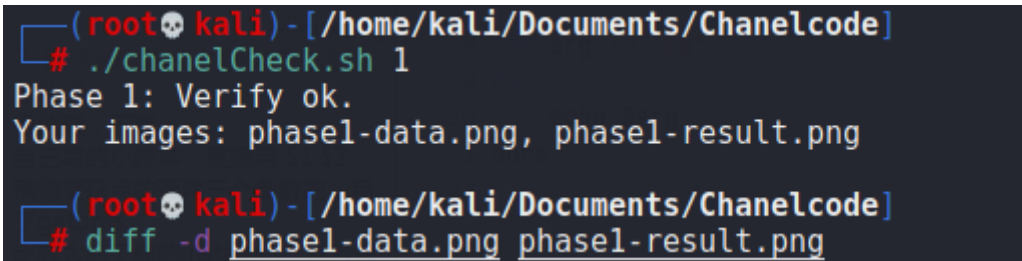
## C5

chanelCheck.sh:

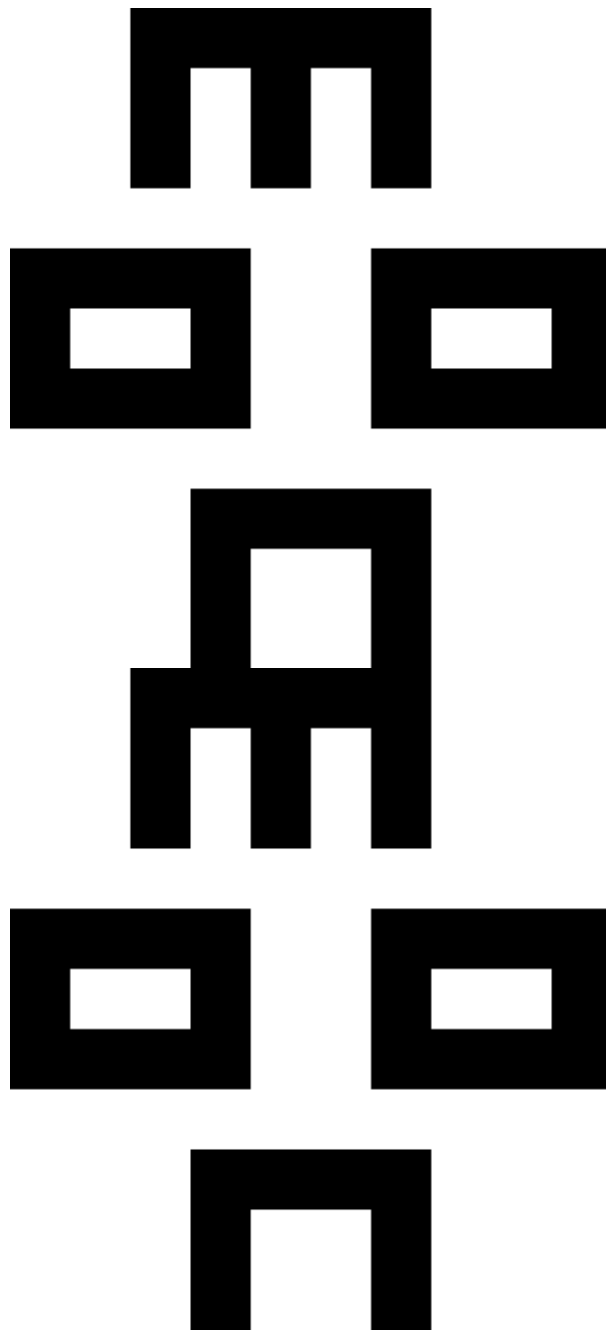
```
./lab2_program_linux_amd64.run -mode=source -phase=$1 | python3 MyCode/ham_io.py  
-e | ./lab2_program_linux_amd64.run -mode=channel -phase=$1 | python3  
MyCode/ham_io.py -d | ./lab2_program_linux_amd64.run -mode=verify -phase=$1
```

### C5.1

运行结果：



```
(root@kali) - [/home/kali/Documents/Chanelcode]  
# ./chanelCheck.sh 1  
Phase 1: Verify ok.  
Your images: phase1-data.png, phase1-result.png  
  
(root@kali) - [/home/kali/Documents/Chanelcode]  
# diff -d phase1-data.png phase1-result.png
```

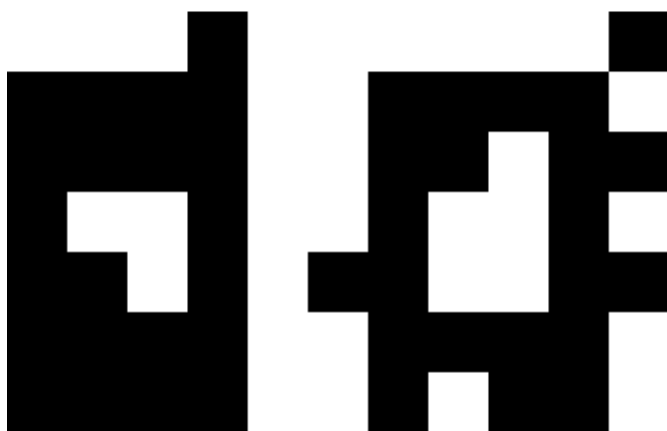
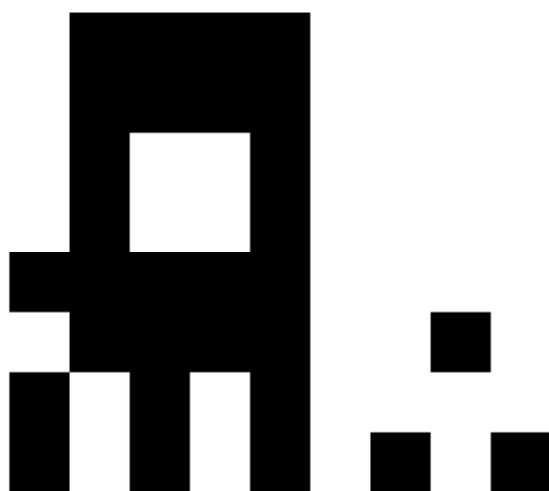


信源数据和解码后的图片相同无差异。噪声信道每15比特中有一个比特出错，而汉明码可以纠一个比特的错误，因此信道解码之后结果与信源输入相同

## C5.2

```
(root@kali) - [/home/kali/Documents/Chanelcode]
# ./chanelCheck.sh 2
Phase 2: Verify ok.
Your images: phase2-data.png, phase2-result.png

(root@kali) - [/home/kali/Documents/Chanelcode]
# diff -d phase2-data.png phase2-result.png
Binary files phase2-data.png and phase2-result.png differ
```



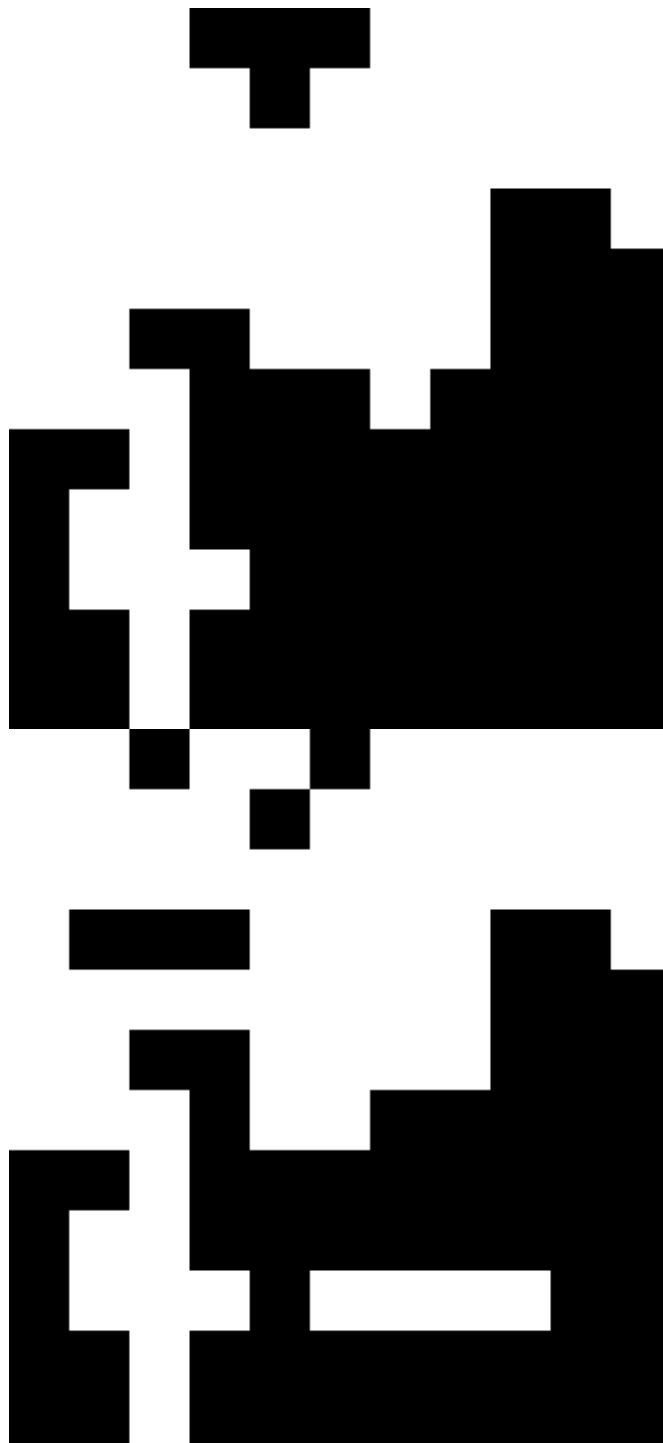
信源数据和解码后图片有差异。因为噪声信道对每30比特的后15比特有2个比特出错，而汉明码只能纠正一个比特的错误，因此信道解码后的结果与信源数据不尽相同。

每30比特中前15比特分组的解码是正确的

### C5.3

```
(root@kali)-[/home/kali/Documents/Chanelcode]
# ./chanelCheck.sh 3
Phase 3: Verify ok.
Your images: phase3-data.png, phase3-result.png

(root@kali)-[/home/kali/Documents/Chanelcode]
# diff -d phase3-data.png phase3-result.png
Binary files phase3-data.png and phase3-result.png differ
```



有差异，每45比特中前15比特有连续的3比特出错，汉明码无法正确纠错。每15比特中间15比特与后15比特通过信道没有出错，解码正确

## OP4

### OP4.1

只考虑最多发生两处错误

由 (8,4) 扩展汉明码编码方法知,  $p = p_0 + p_1 + d_0 + p_2 + d_1 + d_2 + d_3$

- 若  $p + p_0 + p_1 + d_0 + p_2 + d_1 + d_2 + d_3 = 1$ , 则是其中一位出现错误
  - 考察 (7,4) 汉明码部分, 若未出现错误, 则为  $p$  发生错误, 可以直接纠正以及提取数据位
  - 若 (7,4) 汉明码部分检出错误, 因为由拓展汉明码奇偶校验知只有一处错误, 即为 (7,4) 汉明码检出的错误位置, 可以翻转纠错。
- 若  $p + p_0 + p_1 + d_0 + p_2 + d_1 + d_2 + d_3 = 1$ , 则可能没有错误或者两处出错
  - 考察 (7,4) 汉明码部分, 若未出现错误, 则必然是没有错误
  - 若 (7,4) 汉明码部分检出错误, 则可以判断是检测出两个错误, 但是由于不知道具体出错位置, 无法纠错

### OP4.2

部分截图：

```
011111000110111 contains two error, refuse to decode
000010011111111 contains two error, refuse to decode
010000011111111 contains two error, refuse to decode
000110011111111 contains two error, refuse to decode
011100011111110 contains two error, refuse to decode
010111111111111 contains two error, refuse to decode
001011011011111 contains two error, refuse to decode
110100011111111 contains two error, refuse to decode
001100111011111 contains two error, refuse to decode
011011101111111 contains two error, refuse to decode
111111111011111 contains two error, refuse to decode
2048 points checked, decode test pass!!!
```

检测方法：对每个解码的输入，随机产生0-2个错误，与码表对比解码结果，其中两个错误检测到时拒绝解码

具体检测代码：



```

#this part is used to check decode and error detect
with open('hamming_16_11.txt','r') as f:
    checkpoint = 0
    if f:
        with type 8 not holding a Gdk[
18):for line in f:
    f: 10:48:32:685: Event with type 8 not holding a Gdk[
        codeRes = line.split(' ')[1].replace('\n','')
        errorNum = randint(0,2)
        codeRes = list(codeRes)
        if errorNum == 1:
            errorP = randint(0,15)
            codeRes[errorP] = err[codeRes[errorP]]
        if errorNum == 2:
            errorP1 = randint(0,15)
            errorP2 = randint(0,15)
            codeRes[errorP1] = err[codeRes[errorP1]]
            codeRes[errorP2] = err[codeRes[errorP2]]
        codeRes = ''.join(codeRes)
        ham.updateCodeRes(codeRes)
        expectedAns = line.split(',')[0]
        ans = ham.decode()
        if not ans == expectedAns:
            if not ans is None:
                print('check failed in {} -> {},{} expected'.format(codeRes,ans,expectedAns))
                exit(-1)
            checkpoint += 1
print('%d points checked, decode test pass!!!' % checkpoint)

```

```

ans = ham.decode()
if not ans == exp
if not ans is
    print('ch
    exit(-1)
    checkpoint += 1
    print('%d points checked

```